**Ejercicio 1:**

1. 1. **Vim**. Es el más potente de los editores por línea de comandos. Tanto es así, que muchos desarrolladores usan únicamente Vim en su día a día, obteniendo básicamente el mismo nivel de utilidades que muchos editores de interfaz gráfica. Entre sus capacidades más avanzadas nos ofrece coloreado de sintaxis para múltiples lenguajes y tipos de archivo de configuración, capacidad para dividir la pantalla en varios paneles, para editar varios archivos a la vez o acceder a utilidades como el explorador de archivos, undo/redo de manera ilimitada, etc.

El problema de Vim para muchos profesionales que comienzan a usarlo es gestionar sus modos de trabajo y los comandos para realizar las acciones. De todos modos, aunque pueda resultar un poco farragoso inicialmente, con un poco de práctica seremos capaces de obtener prestaciones suficientes como para merezca la pena el esfuerzo de adaptación.

2. Vi. Es el editor precursor de Vim. Aunque es una alternativa más clásica raramente se usa, pues Vim te ofrece un conjunto de utilidades adicionales que suelen ser de agradecer. En cuanto a dificultad ambos editores son igual de complejos, porque comparten los mismos mecanismos de trabajo.

3. Nano. Este es el editor de preferencia de todos aquellos que no se han conseguido adaptar a Vim, o aquellos que, conociendo Vim, prefieren editar el código sin tantas complicaciones. También dispone de coloreado de sintaxis y otras ayudas clave para la programación. Nano permite editar el archivo según se abre el fichero, sin tener que lanzar comandos como en Vim, y ofrece una barra de ayuda siempre visible

b) Editor de texto es un programa informático que permite crear y modificar archivos digitales compuestos únicamente por textos sin formato. En cambio, el comando **CAT** imprimirá por pantalla el contenido del fichero sin ningún tipo de paginación ni posibilidad de modificarlo. Básicamente concatena archivos o la salida estándar en la salida estándar. **MORE** permite visualizar por pantalla el contenido de un fichero de texto, con la diferencia con el anterior de que ‘more’ página los resultados. Primero mostrará por pantalla todo lo que se pueda visualizar sin hacer scroll y después, pulsando la tecla espacio avanzará de igual modo por el fichero. **LESS** es el más completo de los tres, pues puede hacer todo lo que hace ‘more’ añadiendo mayor capacidad de navegación por el fichero (avanzar y retroceder) además de que sus comandos están basados en el editor ‘vi’, del cual se diferencia en que no tiene que leer todo el contenido del fichero antes de ser abierto. Tiene una gran cantidad de opciones y parámetros, como siempre lo recomendable:

**MODOS DE OPERACIÓN VI:**

**Modo de comandos**: podemos desplazarnos dentro de un archivo y efectuar operaciones de edición como buscar texto, eliminar texto, modificar texto, etc. vi suele iniciarse en modo de comandos.

**Modo insertar**: podemos tipear texto nuevo en el punto de inserción de un archivo. Para volver al modo de comandos, presione la tecla Esc (Escape).

**Ejercicio 2:**

1. Paso 0: se ejecuta el código de la BIOS. Paso 1: el HW lee el sector de arranque. Paso 2: se carga el gestor de arranque. Paso 3: se carga el kernel. Paso 4: se monta el sistema de archivos raíz. Paso 5: se ejecuta el proceso init (init es el primer proceso que se ejecuta durante el inicio del sistema en modo usuario (userland), luego de la carga del kernel) Paso 6: lee el /etc/inittab. Paso 7: ejecuta los scripts (instrucciones escritas en código que sirven para ejecutar diversas funciones dentro de un programa) apuntados por el runlevel 1 (se refiere al modo de operación en los sistemas operativos que implementan el estilo de sistema de arranque de iniciación). Paso 8: el final del runlevel 1 le indica que vaya al runlevel por defecto. Paso 9: ejecuta los scripts apuntados por el runlevel por defecto. Paso 10: el sistema está listo para usarse
2. El proceso **init** es el primer proceso en ejecución tras la carga del núcleo y el que a su vez genera todos los demás procesos. Es ejecutado como **demonio** (Se denomina «demonio» o «daemon» a un programa no interactivo (es decir, que el usuario no puede controlar directamente) que se encarga de procesos del sistema en un segundo plano). Su **objetivo** es iniciar y parar todos los servicios, ejecutará las diferentes tareas iniciales necesarias para el funcionamiento de Linux
3. **Pstree** muestra los procesos en ejecución en Linux
4. Los **RunLevels** es cada uno de los estados de ejecución en que se puede encontrar el sistema Linux. Existen 7 niveles: 0- apagado, 1- monousuario (solo usuario root, no es necesaria la contraseña, sirve para analizar y reparar problemas), 2- multiusuario sin soporte de red, 3- multiusuario con soporte de red, 4- como el runlevel 3, pero no se suele usar, 5- multiusuario en modo gráfico, 6- reinicio. Su objetivo es ofrecer distintos tipos de servicios que cada uno tiene asignado
5. Se refiere al modo de operación en los sistemas operativos que implementan el estilo de sistema de arranque de iniciación. Una vez que el kernel de Linux arrancó el init lee la carpeta /etc/inittab que determina el comportamiento de cada nivel. Excepto si el usuario especifica otro valor como parámetro del kernel de arranque, el sistema intentará comentar con el runlevel por defecto
6. El **/etc/inittab** define los puntos importantes del proceso **init**. Indica que le proceso se **init** se iniciará, qué procesos se deben iniciar, supervisar y reiniciar si se terminan, qué acciones se deben realizar cuando el sistema ingresa a un nuevo nivel de ejecución.
7. Hay que ingresar a /etc/inittab y elegir el nivel. Queda predeterminado hasta que sea modificado ya que, al arrancar, siempre elegirá ese nivel
8. El **script RC** es el script superior, el cual es invocado en cada cambio del nivel de ejecución. Ejecuta los scripts de parada del nivel actual y a continuación los scripts de arranque del nuevo. Se almacenan en la dirección /etc/rc.d. Realmente, /etc/rc.d/rc cuando entra en un determinado nivel de ejecución realiza las siguientes acciones: 1. Ejecuta, por orden de nombre, todos los scripts que comienzan por K en el directorio correspondiente al nivel, utilizando como argumento para dicho script la opción stop. 2. Ejecuta, por orden de nombre, todos los scripts que comienzan por S en el directorio correspondiente al nivel, utilizando como argumento para dicho script la opción start.
9. **Insserv** analiza automáticamente los datos introducidos en la cabecera del script init y guarda los enlaces para los scripts de arranque y parada en los directorios de niveles de ejecución respectivos. Esto permite a cada mantenedor de paquetes especificar en su script de init.d la relación con otros scripts y poder detectar y evitar bucles de dependencias entre scripts así como asegurarse de que todos los scripts se inician en el orden pretendido. El programa insserv que contiene este paquete se debería utilizar con cuidado y junto al paquete sysv-rc, ya que utilizarlo incorrectamente puede llevar a un sistema inarrancable. Mejora la performance del arranque en sistemas multiprocesadores
10. **Upstart** es un remplazo basado en eventos, para realizar tareas durante el arranque del sistema. Trabaja de manera asíncrona supervisando las tareas mientras el sistema está arrancando. También gestiona las tareas y servicios de inicio cuando el sistema arranca y los detiene cuando el sistema se apaga. El upstart es un reemplazo basado en eventos, y no en niveles. Los servicios se pueden levantar o desactivar en respuesta a ciertos eventos, y este procedimiento permite por ejemplo manejar el reinicio de servicios que mueren de forma inesperada. Upstart (programado por Scott James Remnant, trabajador de Canonical

Ltd) opera asíncronamente. Las tareas y servicios son ejecutados ante eventos (arranque del equipo o inserción de un dispositivo USB) definidos como tareas o Jobs. Los jobs se almacenan en el directorio /etc/init. Son scripts en texto plano que definen as acciones a ejecutar. Es compatible con el System V (una versión del SO Linux)

1. Upstart es el reemplazo de SystemV (generalmente incluido en Ubuntu y Fedora, openSUSE y otros). Permite la ejecución de tareas en forma asincrónica como principal diferencia con sysVinit que es estrictamente sincrónico. Las tareas y servicios son ejecutados ante eventos (arranque del equipo o inserción de un dispositivo USB) definidos como tareas o jobs. Los jobs se almacenan en el

directorio /etc/init. Son scripts en texto plano que definen las acciones a ejecutar. El init es el responsable de lanzar las tareas. Es compatible con el System V.

1. –
2. **–**
3. **Systemd** es un conjunto de demonios de administración de sistema, bibliotecas y herramientas diseñadas como una plataforma de administración y configuración centrar para interactuar con el núcleo del SO GNU/Linux
4. El **socket** es una representación abstracta de un punto de comunicación, que permite establecer un canal de comunicación entre dos rutinas o programas. // Consultar respuesta
5. **Cgroups** es una característica del kernel de Linux que limita, explica y aísla el uso de recursos de una colección de procesos

**Ejercicio 3:**

1. /home en cuanto a sus “cosas” que quieran hacer, en cuanto a información personal, son /etc/passwd o /etc/shadow
2. Los usuarios son representados por un identificador de usuario, normalmente abreviado como **UID** o User ID. En sistemas tipo Unix el identificador de grupo, abreviado como **GID** o Group ID, es un identificador que permite organizar a los usuarios por grupos

Se supone que no, pero hay un caso excepcional, el UID = 0 (sería el root) Pueden

llegar a haber más de un root, y coexisten, pero son casos excepcionales, se supone

que no puede haber varios usuarios con un mismo ID.

1. El usuario root en GNU/Linux es el usuario que tiene acceso administrativo al sistema. Los usuarios normales no tienen este acceso por razones de seguridad. Tiene UID 0
2. // Hacer

* **USERADD:** permite crear cuentas de usuario. Por medio de parámetros podrán o no tener: nombre de la cuenta, id de usuario, id de grupo, nombre de la persona o servicio, directorio personal o home, Shell a usar

*Useradd asolano:* crea el usuario pelado, sin contraseña, sin interprete de comandos y sin directorio de usuario

*Useradd -m -d /home/depruebas -s /bin/bash:* usuario depruebas creado con directorio de trabajo /home/depruebas e interprete bash /bin/bash, solo le faltaría la contraseña, con el comando **passwd**

*Useradd -G sudo, adm syslog depruebas* añade el usuario depruebas a los grupos, sudo, adm, syslog

* **USERMOD**: permite modificar datos de los usuarios/
  + -a: agrega al usuario a los grupos complementarios
  + -c,- -comment COMENTARIO: comentario será el nuevo valor del campo de comentario en el archivo /etc/passwd
  + -d, - -home HOME\_DIR, el cual será el nuevo valor para el directorio de inicio del usuario. Si se utiliza conjuntamente con la opción -m, el contenido del directorio actual se moverá al nuevo directorio de inicio
  + -e, -expiredate “”: establece la fecha en la que expira la cuenta de usuario. Se especifica en EXPIRE\_DATE que deberá seguir el formato de AAAA-MM-DD
  + -f -inactive INACTIVE: en inactive se establece el número de días que transcurrirán desde que expira una contraseña hasta que la cuenta se deshabilita permanentemente. UN valor de 0 deshabilita la cuenta tan pronto como la contraseña ha expirado y un valor -14 deshabilita la función
  + -g -gid GROUP: en group se establece el nombre del grupo o el identificador de grupo GID del nuevo grupo primario del usuario. El grupo debe existir previamente antes de establecerlo
  + -l -login NEW\_LOGIN: permite cambiar el nombre de cuenta del usuario, probablemente el nombre del directorio de inicio del usuario deberá cambiarse manualmente para que tenga el nuevo nombre de la cuenta
  + -L -lock: bloquea la contraseña del usuario. Esto pone un ¡ delante de la contraseña cifrada en el archivo /etc/shadow, desactivando efectivamente la contraseña
  + -m, -move-home: mueve el contenido del directorio de inicio del usuario a la nueva ubicación
  + -o -non-unique: cuando esta opción se usa conjuntamente con la opción -u, permite cambiar el identificador de usuario a un valor no único
  + -p -passdowrd PASSWORD: establece una nueva contraseña de usuario
  + -s -shell: permite establecer una nueva Shell de inicio de sesión del usuario.
  + -u -uid: establece el nuevo valor número del identificador de usuario
  + -z -selinux-user: establece el usuario de SELINUX para el inicio de sesión del usuario. El valor predeterminado es dejar este campo en blanco, lo que hace que el sistema seleccione el usuario SELINUX predeterminado
* **USERDEL:** es un comando para eliminar usuarias. Buscará los archivos de la cuenta del sistema como /etc/contraseña y /etc/group. Luego, eliminará todas las entradas relacionadas con el nombre de usuario. Si queremos eliminar un usuario debemos hacerlo con privilegio del usuario root. En -r eliminará el usuario doméstico y la cola del correo del usuario. -F fuerza la eliminación del usuario, incluso cuando el usuario aun inicia sesión en el sistema Linux
* **SU:** permite usar el interprete de comandos de otro usuario sin necesidad de cerrar la sesión. Comúnmente se usa para obtener permisos de root para operaciones administrativas, sin tener que salir y reentrar al sistema.
* **GROUPADD:** crea un nuevo grupo usando las opciones específicas en la línea de comando con los valores predeterminados especificados en el archivo. Ej: si quiero crear el grupo mygroup debo ejecutar: groupadd mygroup. El comando agrega una entrada para el nuevo grupo a los /etc/group y /etc/gshadow. De forma predeterminada, cuando se crea un nuevo grupo, el sistema asigna el siguiente GID disponible del rango de ID de grupo especificado en el archivo login.defs, usando -g para crear un grupo con un GID especifico. Entonces: group -g 1010 mygroup
* **WHO:** el comando muestra rápidamente los usuarios que están autenticados en el sistema. Usando -h añadirá encabezados a las columnas según corresponda en la salida de la ejecución. Who -b/who –boot nos brinda información sobre el tiempo desde que se inició el sistema. Who -q muestra los usuarios que se encuentran logeados. Who -t nos permite agregar un carácter que indique el estado de la línea de la terminal. Who -all permite ejecutar todas las opciones de forma simultanea
* **GROUPDEL:** se usa para eliminar un grupo. Ej: groupdel migrupo
* **PASSWD:** utilizado para cambiar la contraseña del usuario que se pasa como argumento. -a informa el estado de la contraseña de todas las cuentas, -d borra la contraseña para la cuenta indicada, -e fuerza a que la contraseña de la cuenta caduque, -h muestra este mensaje de ayuda y termina, -k cambia la contraseña solo si ha caducado, -l bloquea la contraseña de la cuenta indicada, -q modo silencioso, -S informa el estado de la contraseña de la cuenta indicada, -u desbloquea la contraseña de la cuenta indicada, -i INACTIVO establece la contraseña inactiva después de caducar a INACTIVO, -w DIAS\_AVISO establece el aviso de caducidad a DIAS\_AVISO, -n DIAS\_MIN establece el número mínimo de días antes de que se cambie la contraseña a DIAS\_MIN, -x DIAS\_MAX establece el número máximo de días antes de cambiar la contraseña a DIAS\_MAX

**Ejercicio 4:**

1. En Linux todo archivo y directorio tiene 3 niveles de permisos de acceso: los que aplican al propietario del archivo, los que se aplican al grupo que tiene el archivo y los que se aplican a todos los usuarios del sistema. Cuando listamos un directorio con ls -1 podemos ver los permisos. Tipo lectura: permite a un cierto usuario ver el contenido de un archivo. SI es un directorio, el contenido del mismo, escritura: permite modificar un archivo, ejecución: si existe algo que ejecutar, lo hará con este permiso.
2. **CHMOD:** usado para cambiar los permisos de archivos o directorios. CHMOD= modo de cambio y se usa para definir la forma en que se puede acceder a un archivo. -f: no visualiza posibles mensajes de error que puedan ocurrir debido a conflictos en la asignación de permisos, -v lista los ficheros y directorios a los que se les va aplicando el comando a medida que el mismo se ejecuta, -R aplica el comando chmod recursivamente a todos los ficheros y de los subdirectorios. PERMISOS BASICOS: r- lectura, w- escritura x- ejecución

**CHOWN:** permite cambiar el propietario de un archivo o directorio. Puede especificarse tanto el nombre de un usuario, así como el UID y GID.

**CHGRP**: cambia el grupo propietario de un fichero/directorio.

1. R: read: solo lectura = 4, W: write: solo escritura = 2, X: execute: solo ejecución = 1
2. Existe la posibilidad, si es que el usuario puede utilizar el comando su, o sudo. Si no, sin poseer los permisos necesarios no puede acceder al archivo. Sólo root.
3. **Full path name:** Un nombre de vía de acceso completo empieza en el directorio raíz y efectúa un rastreo de todos los directorios que quedan por debajo hasta llegar al archivo y directorio de destino. Por ejemplo, /etc/uucp/Devices hace referencia al archivo Devices del directorio uucp del directorio raíz etc. Para indicar un directorio raíz, siempre es preciso especificar delante el carácter de barra inclinada (/). Separe siempre los elementos de la vía de acceso mediante el carácter de barra inclinada (/).

**Relative path name:** El nombre de vía de acceso relativo sólo contiene los directorios que dependen del directorio actual. Por ejemplo, si el directorio actual es /usr/bin y el directorio de destino es /usr/bin/reports, escriba el nombre de vía de acceso relativo reports (sin la

barra inclinada inicial).

1. El comando pwd viene de “print working directory” y cuando se lo escribe, le está preguntando al sistema que muestre su ubicación actual. Puedo acceder al sistema con /home o cd. Sí, se podría acceder a otros directorios como /etc
2. **CD:** es un comando que significa “cambiar directorio”, usado para cambiar el directorio. -L siga los enlaces simbólicos. -P no seguir enlaces simbólicos, cuando se especifique esta opción e intente navegar a un enlace simbólico que apunta a un directorio, cd se convertirá en el directorio. Enlace simbólico= indica un acceso a un directorio o fichero que se encuentra en un lugar distinto dentro de la estructura de directorios. Si quiero cambiar a un nivel menos debo escribir cd../ o cd../../ para volver a dos niveles menos

**UMOUNT:** elimina el anclaje entre el dispositivo especial y el punto de montaje. Si tenemos una unidad de CD montada no podremos extraer el CD hasta que lo desmontemos. Si estamos trabajando con una unidad de disquete es fundamental desmontarla antes de extraer el disco.

**MKDIR:** comando usado para crear un nuevo subdirectorio o carpeta del sistema de archivos. -m establece los permisos de los directorios creados, -p crea los directorios padres que falten para cada argumento directorio

**DU:** informa cuánto espacio en disco ocupa un archivo o directorio, -a muestra el uso de espacio de cada archivo, -k escribe el tamaño de los archivos en unidades de 1024 octetos en vez de las unidades de 512 octetos por defecto, -s en vez de la salida por defecto, informa solo de la suma total de cada uno de los discos especificados, -L procesa los enlaces simbólicos usando el archivo o directorio al que el enlace simbólico referencia, -c cuando evalúa los tamaños de archivo, evalúa solo esos archivos que tengan el mismo dispositivo que el archivo especificado en el operando

**RMDIR:** permite eliminar un directorio. <unidad> especifica la unidad, <ruta> especifica la ruta, S/ quita todos los directorios y archivos contenidos en el directorio especificado, además de eliminar dicho directorio, /Q no pide confirmación a la hora de liminar un árbol de directorios con /S, /? Muestra información de ayuda acerca de RMDIR

**DF:** nos permite conocer la cantidad de espacio libre y utilizado por nuestro sistema de archivos en nuestras unidades de almacenamiento

**MOUNT:** se utiliza para montar dispositivos y particiones para su uso por el sistema operativo. Montar es hacer que el SO proyecte el contenido de ese dispositivo en un enlace lógico. Auto/ noauto: el dispositivo será montado automáticamente durante el inicio o en caso de que el comando mount -a sea ejecutado, auto es el valor por defecto, dev/ nodev: interpretar/ no interpretar dispositivos especiales de bloques en el sistema de archivos, exec.noexec: permite ejecutar binarios que están en la partición, mientras que noexec lo impide. Ro: montar solo para lectura, rw: montar el sistema de archivos para lectura/ escritura, sync/ async: indica la manera en que debe realizar la entrada y salida del sistema de archivos, sync especifica que se realice de manera sincrónica, suid/ nosuid: permite/ bloquea la operación sobre los bits suid y sgid, user/nouser: permite a cualquier usuario montar el sistema de archivos

**LN:** crea enlaces entre ficheros

**LS:** lista el contenido de un determinado directorio o, si no le indicamos el directorio, lista el contenido del directorio actual.

**PWD:** utilizado para imprimir el nombre del directorio actual

**CD:** usado para cambiar el directorio de trabajo

**MV:** mueve o renombre archivos o directorios del sistema de archivos. -b: no acepta un método de copia de seguridad, -f: sobrescribe los archivos existentes sin preguntar, -i: usado para configurar que pregunte antes de sobrescribir un archivo existente, -n: con esta opción nunca sobrescriba ningún archivo existente, -S: el sufijo del nombre de archivo que se utilizara para todos los archivos de copia de seguridad, -t: mueve todas las fuentes al destino del directorio, -u: le indicamos que no sobrescriba los archivos si son mas nuevos, -v: proporciona una salida detallada, imprima el nombre de cada archivo movido

**Ejercicio 5**:

1. Un proceso es una instancia de un programa en ejecución. Programas y procesos no son iguales. En un SO multitarea, múltiples instancias de un programa pueden ejecutarse simultáneamente. Cada instancia es un proceso separado. Prácticamente todo lo que está ejecutando en el sistema en cualquier momento, es un proceso. **PID:** es una abreviatura de process ID, o sea ID del proceso. El cual es un numero entero usado por el kernel para identificar un proceso de forma univoca. **PPID:** es el PID del padre del proceso. Un proceso padre en Linux tiene la capacidad de generar uno o varios procesos hijo, pero el proceso hijo único solo puede pertenecer a un proceso principal. Todos los procesos tienen estos atributos
2. Ps, pstree y top. Las opciones más importantes y utilizadas de este comando son: -a para mostrar los procesos de todos los terminales, -u para mostrar el usuario al que pertenece el proceso y la hora de inicio, y -x para mostrar procesos que no están controlados por ningún terminal. Suelen usarse combinadas para tener una visión global de los procesos que están en ejecutan. Otros dos comandos útiles a la hora de visualizar los procesos son: pstree, que nos muestra los procesos en una estructura de árbol; y top.
3. **Background:** se utiliza para nombrar a todos aquellos procesos o rutinas de ejecución que se encuentran en segundo plano. Esto implica que el proceso se está llevando a cabo con una prioridad baja y no siempre tiene la CPU de forma secuencial ejecutando su código. **Foreground:** significa que se esta ejecutando en primer plano, un programa en foreground lanzado desde un terminal monopoliza dicho terminal, por lo que en principio, no podremos ejecutar ningún otro programa a la vez
4. Foreground: Los procesos se inician de este modo simplemente escribiendo el comando en la línea de comandos. Background: Los procesos se inician de este modo escribiendo un & al final de la línea del comando. También se puede colocar un proceso en background a través de la utilización de las teclas Ctrl+z seguido del comando bg. Para retornar el proceso al shell principal, utilizamos el comando fg [N\*], de esta forma el proceso vuelve a shell inicial. N\*: número de orden de entrada en el sistema
5. Los pipes son una forma de redirección que se utiliza para enviar la salida de un programa a la entrada de otro. Un pipe es un par de "archivos inexistentes", que tienen la cualidad de que lo que se escribe en uno se lee en el otro. Este mecanismo nos permite pasar la salida de un comando a otro. Para ello se usa la sintaxis: <comando1> | <comando2>. Con esto, la salida de comando1 será la

entrada de comando2. Vamos a ver unos ejemplos: $ rpm -qa | grep <nombre paquete> El primero de los dos comandos nos haría una lista de todos los paquetes instalados. Imaginemos que sólo queremos saber si tenemos instalado uno en concreto. Con el segundo comando limitamos la salida a los paquetes que en el nombre que contengan el patrón que especificamos en <nombre paquete>. Por ejemplo, para saber si tenemos instalado algún paquete llamado glibc haríamos: $ rpm -qa | grep "glibc" grep es un parseador de expresiones regulares, es decir, le damos un patrón y un fichero (o introducimos lo que sea por consola, o lo pasamos con un pipe) y de ese texto nos devuelve sólo lo que coincide con el patrón. Otro ejemplo útil seria, por ejemplo, cuando queremos saber el PID de un proceso. En vez de mostrarlos todos y tener que buscarlo podríamos hacer: $ ps -e | grep <nombre proceso> y así nos mostraría sólo las líneas que contuvieran <nombre proceso> (es decir, limitaríamos la salida al proceso que queremos).

1. Las redirecciones consisten en trasladar información de un tipo a otro como de error estándar a la salida estándar o de la salida estándar a la entrada estándar, usando el símbolo >. En Linux, al final todo es tratado como si fuera un fichero y como tal, tenemos descriptores de fichero para aquellos puntos donde queramos acceder. Hay unos descriptores de fichero por defecto: 0: Entrada estándar (normalmente el teclado). 1: Salida estándar (normalmente la consola). 2: Salida de error. Para redirigir las salidas utilizaremos el descriptor de fichero seguido del símbolo '>' (o < si redirigimos la entrada hacia un comando). Veamos unos ejemplos: $ ls -l >fichero

Guarda la salida de ls -l en fichero. Si no existe lo crea, y si existe lo sobrescribe. $ ls -l >>fichero Añade la salida del comando a fichero. Si no existe lo crea, y si existe, lo añade al final. $ ls -l 2>fichero Si hay algún error, lo guarda en fichero (podría salir un error si no tuviéramos permiso de lectura en el directorio). Es importante ver que si no se especifica el descriptor de fichero se asume que se redirige la salida estándar. En el caso del operador < se redirige la entrada estándar, es decir, el contenido del fichero que especificáramos, se pasaría como parámetro al comando. Si quisiéramos redirigir todas las salidas a la vez hacia un mismo fichero, podríamos utilizar >&. Además, con el carácter & podemos redirigir salidas de un tipo hacia otras, por ejemplo, si quisiéramos redirigir la salida de error hacia la salida estándar podríamos indicarlo con: 2>&1. Es importante tener en cuenta que el orden de las redirecciones es significativo: se ejecutarán de izquierda a derecha.

1. Es un comando usado para enviar mensajes sencillos a los procesos ejecutándose en el sistema. Por defecto el mensaje que se envía es la señal de terminación, que solicita al proceso limpiar su estado y salir. Es un wrapper alrededor de la llamada al sistema kill(), que

envía señales a procesos o grupos de procesos en el sistema, referenciados por sus IDs de proceso (PIDs) o IDs de grupo de procesos (PGIDs). kill se ofrece siempre como programa independiente, pero la mayoría de las shells tienen intrínsecamente comandos kill que pueden diferir levemente de ella. Pueden enviarse numerosas señales. La señal por defecto es SIGTERM. Los programas que cazan esta señal pueden limpiar su estado (como guardar datos de la configuración a un archivo) antes de terminarse. Para los programas que no capturan esta señal, se utiliza una gestión por defecto. En otras ocasiones, un proceso capaz de capturar la señal puede quedar en un estado que le impide manejarla. Todas las señales excepto SIGKILL y SIGSTOP pueden ser interceptadas por el proceso, esto es, tener una función especial que es llamada cuando el programa recibe tales señales. Sin embargo, SIGKILL y SIGSTOP sólo las ve el kernel, lo que provee formas seguras de controlar la ejecución de los procesos. SIGKILL finaliza el proceso, y SIGSTOP lo pone en pausa hasta que se reciba una señal SIGCONT. Unix cuenta con mecanismos de seguridad para evitar que usuarios no autorizados

puedan finalizar otros procesos. Básicamente, para que un proceso pueda enviar una señal a otro, deben tener el mismo propietario, o ser enviada por el supe usuario. Las señales disponibles tienen distintos nombres, asignados a determinados números. El número de las señales puede cambiar entre distintas implementaciones de Unix. SIGKILL suele tener el número 9 y SIGTERM el 15.

1. **PS:** “process status”permite visualizar el estado de un proceso. -A: muestra todos los procesos, -a muestra los procesos de una tty determinada, -d: muestra todo excepto los lideres de la sesión, -e: muestra todos los procesos (equivalente a -A), T: muestra todos los procesos de la terminal actual, a: muestra todos los procesos de la terminal actual incluyendo los de otros usuarios, g: muestra todos los procesos incluyendo grupos lideres, r: muestra solamente procesos corriendo, x: muestra procesos en un estilo BSD, fL visualiza los parámetros con los que se levantó el proceso.

**KILL:** Permite interactuar con cualquier proceso mandando señales. Kill <pid> termina un proceso y Kill -9 <pid> fuerza a terminar un proceso en caso de que la anterior opción falle.

**PSTREE:** ps en forma de árbol

**KILLALL:** es mas sencillo que kill, con tan solo conocer el nombre del proceso que queremos matar, es suficiente.

**TOP:** nos da información acerca del uso de la CPU, de la memoria y de los procesos en ejecución etc en tiempo real. D: podemos cambiar el tiempo de refresco, pulsando n y a continuación un número, podemos ver el número de procesos con más consumo que queramos.

**NICE:** nos permite modificar la prioridad de un proceso frente al resto dentro del sistema. Nice -n -5 find: el valor de la prioridad del proceso find decrece en 5, aumentando su prioridad

**Ejercicio 6:**

1. Empaquetar es juntar varios archivos en un solo fichero.
2. El tamaño del archivo empaquetado es menor al tamaño de los 2 archivos por separado, sin empaquetar.
3. tar –cvzf mi\_archivo\_comprimido.tgz /home/fabrizio/archivo1o2o3o4
4. No, debo comprimir de a uno, aunque sea engorroso, es así. Lo que sí puedo es empaquetar, primero, varios archivos en uno, y a esto comprimirlo, con lo cual estaría comprimiendo un conjunto de archivos, pero antes empaquetándolo, como dijimos.
5. **TAR:** es un comando para comprimir archivos. -c crea un nuevo archivo,tar, -v: muestra una descripción detallada del proceso de la compresión, -f: nombre del archivo, -z: comprensión gzip, -j: comprensión bzip2, -CL extraer archivos en un directorio diferente, -x: extraer el archivo, -r: actualizar o agregar un archivo o directorio en un archivo .tar existente

**GREP:** toma una expresión regular de la línea de comandos, lee la entrada estándar o una lista de archivos e imprime las líneas que contengan coincidencias para la expresión regular.

**GZIP:** herramienta de comprensión, que se usa para truncar el tamaño de un archivo.

**ZGREP:** busca archivos comprimidos para una expresión.

**WC:** comando usado para realizar diferentes conteos desde la entrada estándar.

**Ejercicio 7:**

**ls -l:** genera archivo prueba que contiene un listado con todos los archivos contenidos en el directorio home. Se redirige la salida estándar de ls mediando > al archivo prueba

**ps >PRUEBA:** genera un archivo de nombre PREUBA que contiene un listado de los procesos en ejecución en el directorio home del usuario. Se redirige la salida estándar mediante >

**chmod 710 prueba:** ambia los permisos del archivo prueba a 710 para UGO

**chdown root: root PRUEBA:** se intenta cambiar el propietario del archivo prueba pero la operación no está permitida

**chmod 777 PRUEBA:** cambia los permisos del archivo prueba a 777. Es decir, todos los usuarios pueden leer, escribir y ejecutar archivo.

**Chmod 700/etc/passwd:** intenta cambiar los permisos a 700, pero la operación no está permitida para un usuario que no es root, esto por el archivo que está intentando cambiar

**Passwd root:**  passwd: no debe ver o cambiar la información de la contraseña para root

**Rm prueba:** elimina archivo prueba

**Man /etc/shadow:** permiso denegado, porque “man”no debe recibir una ruta, si hago “man shadow” sí anda

**Find/ -name\*.conf:** lista todos los archivos cuyos nombres terminan con .conf, empezando la búsqueda en el directorio raíz /

**Usermod root -d/home/nwroot -L:**

**Cd/root:** se intenta acceder a la carpeta root

**Rm\*:** borra todos los archivos del directorio donde está posicionado el usuario

**Cp \*/home -R:** intenta copiar el contenido de /etc a home, pero el usuario no tiene los permisos necesarios

**Shutdown:** apagará el equipo

**Ejercicio 8:**

1. Se usará kill 23
2. Pkill init, no hará nada porque el proceso init no puede detenerse
3. Find/-name \* .conf
4. ps > /home/SO/procesos
5. –
6. –
7. Cd /tmp rm\*
8. chown iso2010 /opt/isodata
9. pwd >> /home/iso/donde

**Ejercicio 9:**

1. su root
2. $su useradd savila passwd savila
3. Se modificaron los archivos de etc/shadow y se creó el directorio home/savila
4. Cd /tmp mkdir cursada2021
5. cp /var/log/\* /tmp/cursada2021 {CP copia archivos}
6. chown adibello:users /tmp/cursada2021
7. chmod 723 /directorio/ -R
8. screen su savila
9. hostname
10. ps -e
11. who -q
12. write savila echo “El sistema será apagado, guarde sus trabajos”
13. shutdown

**Ejercicio 10:**

1. mkdir 185443 cd 185443
2. vi LEAME; apreto “i” para ingresar todo lo necesario, escape y :wq.
3. chmod 017 LEAME
4. cd /etc

ls -l, lo cual es posible crearlo ya que es case sensitive

1. Encontrar un archivo en el filesystem: sudo find / -name “archivo” –type f

Encontrar todas las imágenes en el filesystem: sudo find / -name “\*.jpg” –type f

1. find / -name “\*.so” –type f > /home/SO/185443/ejercicio\_f

**Ejercicio 11:**

**mkdir iso** = crea directorio llamado iso

**cd . / iso; ps>f0** = accede al directorio iso y lista los procesos almacenándolos en el archivo llamado f0

**ls > f1**= lista los archivos contenidos en el directorio iso y los almacena en el archivo f1

**cd /** = vuelvo a la raíz

**echo $HOME** = imprime el directorio HOME

**ls −l $**> $HOME/ iso/ ls= guarda los detalles del directorio home en el archivo ls el cual se ubica en el directorio iso

**cd $HOME**; mkdir f2= va a home y crea el directorio f2

**ls −ld f2**= lista los detalles del directorio f2

**chmod 341 f2**= cambia los permisos del directorio f2, escritura y ejecución al dueño, lectura al grupo y ejecución a otros //investigar más sobre chmod

**touch dir**= cambia la fecha del directorio

**cd f2**= accede al directorio f2

**cd ~/ i s o**= vuelve al directorio iso

**pwd >f3**= imprime nombre del directorio iso y lo almacena en f3

**ps | grep ' ps ' | wc −l >> . . / f 2 / f 3**= Guarda en el archivo f3, sin sobreescritura, ubicado en el directorio f2, la cantidad de

líneas que contienen la palabra “ps” entre la lista de procesos.

**chmod 700 . . / f 2 ; cd .** .= Modifica los permisos del directorio f2: total para usuario, nada para grupo y nada

para otros. Vuelve a home.

**find. −name etc /passwd**=

**find / −name etc /passwd**= permiso rechazado

**mkdir ejercicio5**= crea el directorio ejercicio5

**Ejercicio 12:**

1. mv f3 /home/iso

cp f4 /home/iso/dir1/dir11

//no sé

cd ~/usuario mkdir copia

cd ./dir1 cp f11 /home/usuario/copia

cd /home/iso/f1 mv f1 archivo ls -l

chmod 613 archivo

cd ~/usuario cd dir3 mv f3 f3.exe

cd ~/usuario cd dir1 mv f4 f4.exe

chmod 023 f4.exe f3.exe

**Ejercicio 13:**

1. cd /tmp mkdir logs
2. cp /home/root/var/log/ \* /Home/root/tmp/logs
3. tar –cvf misLogs.tar /home/root/tmp/logs
4. tar –cvzf misLogs.tar.gz /home/root/tmp/logs
5. cp /home/root/logs/misLogs.tar /home/root

cp /home/root/logsmisLogs.tar.gz /home/root

1. rmdir logs
2. tar –xvf misLogs.tar > /home/root/directorio1

tar-xvzf misLogs.tar.gz > /home/root/directorio2